1.7 数値のフォーマット　　ver. 0.1

I/O のセクションを終えるにあたって、もう１つだけ重要なトピックスを取りあげたいと思います。それは、print 時の数値のフォーマットについてです。簡単なようで、実は多くの誤解が蔓延しています。指導者が「このフォーマットを使うように」とただ漫然と説明なしで使うことは、多くの誤解と誤った使用法を伝承ゲームのように伝えるだけです。この際、きっちりとその概念と仕組みをおさえてください。

まず、GAUSS 内部では、数値はすべて二倍精度浮動小数点で扱われます。したがって、プログラムの始めのところに format 文を書いたとしても、内部の計算に使われる桁数が変更されるわけではありません。常に、二倍精度浮動小数点で扱われ、有効桁数は 15 桁から 16 桁の間に設定されています。この点は、ＳＡＳなどのメジャーなソフトウエアと同じはずです。（ただし、ライブラリーなどを扱う場合には、そのライブラリーの設定に依存します。ライブラリーとは、GPE2 や maxlik などの呼び出して使う GAUSS 標準の機能ではないものをさします。）format 文は計算の途中の有効桁数には何も影響を与えないことを誤解のないよう覚えておいてください。

GAUSS では、数値を print する場合（画面表示またはファイル出力）、それぞれの数値の入る最大桁数と小数点などを調節することができます。（繰り返しますが、途中計算の有効桁数を調節するという意味ではけしてありません。）出力時の桁数調節には、２つの代表的な方法があります。まず簡便な方法として、print 文にオプションを書き加える方法。もう１つには、format という文を print 文の前のどこかに置いて、本格的に調節する方法があります。

まずは、print 文だけでやってみましょう。桁数を整える以外の、たいていの print 時のフォーマットができてしまいます。文字間隔の概念が複雑ですから縮尺を表示させます。

プログラム

```
new; cls;
a = -10000.0000;
print "----|----1----|----2----|----3----|----4----|----5";
print "----|----|----|----|----|----|----|----|----|----|";
print a;
print /re a;
print /rd a;
print /ro a;
print /rz a;
print "----|----|----|----|----|----|----|----|----|----|";
```

画面結果

```
----|----1----|----2----|----3----|----4----|----5
----|----|----|----|----|----|----|----|----|----|
       -10000.000
-1.00000000e+004
 -10000.00000000
       -10000.000
            -10000
----|----|----|----|----|----|----|----|----|----|
```

（注意：ペーストした際、英語の文章であれば courier、日本語のものであれば明朝とかでなければ、１文字１スペースにはならずに、ずれてしまうことがあります。）

上のような表示になったでしょうか。GAUSS のデフォルトでは、数値は 16 桁の領域が確保されて、原則そこに表示されるということがわかります。まず第 2 番目は、/re のオプションをつけると、右そろえ（ｒ）で指数表示（ｅ）となることがわかります。この場合、－１の１０の４乗倍という意味です。ｅのあとにマイナスのマークがくると１０のマイナス何乗倍ということになります。第 3 番目は、/rd のオプションをつけると右そろえ（ｒ）で十進法（ｄ）で表示するという意味で、小数点以下はデフォルトの８桁となります。（ただし、十進法表示の場合には、桁数はデフォルトの８桁のままですが、小数点以上の数の桁数はその数の桁数に応じて増加して、全体として１６桁の領域を越えることもあり、ｄオプションの場合には全体の桁数の規則は守られません。十進法の表示をあくまでも守るため、全体の桁数は例外的に確定しません。）第４番目と第 1 番目は同じです。これは、GAUSS のデフォルトの設定が、/ro という設定になっているためです。指数表示または十進法表示のどちらか短く表示できる方法に最適化(o)されて表示されます。この場合には、十進法が自動的に選ばれています。そして、全体の桁数は、小数点やマイナスサイン、0. の０の桁を除いて８桁になっています。最後５番目は、/rz オプションで、/ro の機能に加えて、小数点以下にゼロがある場合には、ゼロをなくす（ｚ）という役割を果たします。

　さて、右そろえのｒのところをｌにすれば、当然のことながら、左そろえになります。以下のように、上のプログラムを変更してみてください。（ｌはＬの小文字です。）

プログラム変更箇所

```
print /le a;
print /ld a;
print /lo a;
print /lz a;
```

画面結果

```
----|----1----|----2----|----3----|----4----|----5
----|----|----|----|----|----|----|----|----|----|
       -10000.000
-1.00000000e+004
-10000.00000000
-10000.000
-10000
----|----|----|----|----|----|----|----|----|----|
```

上のようになったでしょうか。ｌは左そろえという意味のオプションです。オプションのない第 1 番目は、デフォルト設定の、右そろえのままです。ｅ、ｄ、ｏ、ｚはすべて右そろえのときの意味と同じです。デフォルトでは、領域は 16 文字スペース分いずれの場合も確保されています。（ただし、ｄの十進表示の場合には、16 文字スペースを超える場合もあります。）16 文字スペースの左に寄って、すべて左そろえになりました。

　通常は、右そろえ（ｒ）のまま出力する場合が大多数ですが、次のように文字列とともに数値を表示する場合には、左そろえ（ｌ）も有効に利用できます。

プログラム

```
new; cls;
a = -1000.0000;
print /lo "a=" a;
```

画面表示

```
a=-1000.0000
```

数値を説明する文字列（この場合は a=）があるときには、デフォルトの右そろえ（ｒ）では、間のびした出力になってしまいます。そんなときには、左そろえ（ｌ）で、文字列のあとに数値を直接書きこめて便利です。注意してほしいことは、ｒやｌ単独ではオプションにはなりえないことです。必ずｅ、ｄ、ｏ、ｚのうちのどれかが添えられて、２文字でオプションになります。

　ｄオプション（十進法表示）の場合、右そろえでも左そろえでも両方とも、極端に小数点以上の部分の桁数が多い場合には、そのすべての桁を表示しなければならないため、全体の文字スペース 16 のデフォルトを超過することがあります。これは、十進法表示では、指数表示と違って、桁数が極端に大きくなってもそのすべての数を表示しなければ数値を表しきれないという根本的な理由からきています。けして、GAUSS が壊れているわけでも

いい加減なわけでもありません。ちゃんとした論理的理由があって、十進法で表しきってしまうということが優先して、デフォルトの桁数が守られないわけです。

プログラム

```
new; cls;
a=-100000000000.001;
print "----|----1----|----2----|----3----|----4----|----5";
print "----|----|----|----|----|----|----|----|----|----|";
print /rd a;
print /ld a;
print "----|----|----|----|----|----|----|----|----|----|";
```

画面結果

```
----|----1----|----2----|----3----|----4----|----5
----|----|----|----|----|----|----|----|----|----|
-100000000000.00101000
-100000000000.00101000
----|----|----|----|----|----|----|----|----|----|
```

rd（右そろえの十進法表示）でも、ld（左そろえの十進法表示）でも、両者ともに、16 桁をはるかに超えてしまいました。依然として、小数以下はデフォルトの８桁なのですが、小数以上の部分をすべて表示しなければならないため、自動的に全体の文字スペースのデフォルトの値 16 は無視されているのです。

ポイント　print 文における桁数の制御の方法には次のものがある

print /re 変数;　　右そろえ指数表示
print /rd 変数;　　右そろえ十進法表示
print /ro 変数;　　右そろえ上のどちらかコンパクト方の正味８桁表示法
print /rz 変数;　　右そろえ/ro に加えて小数点以下最終０の省略
print /le 変数;　　左そろえ指数表示
print /ld 変数;　　左そろえ十進法表示
print /lo 変数;　　左そろえ上のどちらかコンパクト方の正味８桁表示法
print /lz 変数;　　右そろえ/ro に加えて小数点以下最終０の省略
デフォルトは、　print /ro

ポイント　/rd と/ld オプションは 16 桁の文字スペースを超えることもある

さて、数値が行列の場合どうでしょうか。GAUSS では、特にことわらないかぎり、数値は行列として扱われるので、厳密に言えば、上の例も１×１行列の例だったのですが、普通の２×２以上の行列の出力について見ていきましょう。行列の場合、となりの数との間隔、行と行の間隔など考えなければならない問題も出てきます。

プログラム

```
new; cls;
rndseed 999;
a = rndn(3,3);
print "----|----1----|----2----|----3----|----4----|----5";
print "----|----|----|----|----|----|----|----|----|----|";
print a;
print /re a;
print /rd a;
print /rz a;
print "----|----|----|----|----|----|----|----|----|----|";
```

画面結果

```
----|----1----|----2----|----3----|----4----|----5
----|----|----|----|----|----|----|----|----|----|

         1.3220575          1.3315518        -0.45884438
         1.0304805        -0.10084612        -0.28467660
        0.51318499          1.1745187          1.9421390

 1.32205752e+000  1.33155183e+000 -4.58844377e-001
 1.03048047e+000 -1.00846115e-001 -2.84676596e-001
 5.13184990e-001  1.17451874e+000  1.94213898e+000

        1.32205752         1.33155183        -0.45884438
        1.03048047        -0.10084612        -0.28467660
        0.51318499         1.17451874         1.94213898

         1.3220575          1.3315518        -0.45884438
         1.0304805        -0.10084612         -0.2846766
        0.51318499          1.1745187           1.942139
----|----|----|----|----|----|----|----|----|----|
```

この例は、また右そろえ（ｒ）に戻り出力を調節しています。上の例からわかることは、デフォルトではすべて 16 文字スペースがそれぞれの行列の値に確保されていて、それぞれの行において、値の間に１スペースあいているということです。ですから、１列目は 16 文字スペースのところで右そろえになっており、2 列目 16×２＋１＝33 文字スペースのところで右そろえになっています。３列目は、同様に、16×３＋２＝50 文字スペースのところで右そろえになっています。これは左そろえ（ｌ）の場合でも、同じことがあてはまります。繰り返しになりますが、若干細かいところで規則があるので、個々の例を説明していきます。以前でできましたように、standard normal な分布にしたがう同じ乱数を再現するために、シードを設定します。仮に、ここでは 999 としましょう。３行３列、シードの値を再現する乱数を発生させます。まず、第１番目の出力は、デフォルトのままのものです。厳密には、デフォルトは/ro の設定になっています。これは、最もコンパクトな数値表示方法を十進法表示と指数表示のうちから自動選択されるオプションです。この場合には、十進法表示が選択されています。しかしながら、第 3 番目の十進法表示とは若干桁数に違いが見られます。このデフォルトの/ro は、小数点、マイナスのサインに加えて、0.なにかという小数の場合の０の桁も除いた、全体の桁が８桁に定まっています。したがって、第３列目の十進法表示にくらべて、各行列の要素の数値がそろっていません。正味８桁ということになっているため、こういうことがおこっています。第２番目の/re のオプションでは、数値と数値の間にあるスペースがより鮮明にわかるかと思われます。/re も/le も通常は、マイナスのサインの入る最初のスペース以外すべての桁を使いきるので、数値と数値の間の１スペース分がきわだちます。見てわかるように、その１列目はマイナスのサインがないにもかかわらず、マイナスのスペースがリザーブされて、第２文字スペースから始まっています。マイナスがついたときには、ここにマイナスが入ります。表示の桁をそろえる意味から言えば、/re オプションが最も安定していて扱いやすい方法でしょう。5.13184990e-100 というのは、5.13184990 の 10 のマイナス１乗、すなわち 10 分の１という意味です。全体でマイナスなどの符合もすべて含めて 16 文字スペースを利用し、小数点以下は８桁がデフォルト設定です。第 3 番目の/rd は十進法表示で、同様に小数点以下は、８桁がデフォルト設定です。なお、全体の桁数は、小数点以上の部分が十分に大きければ、16 桁の枠を超えることもあります。最後４番目の/rz オプションは、基本的には、第１番目の/ro のデフォルトと同じですが、小数点部分の最後に０が１つ以上くると、その０が省略されて表示されるものです。そういうわけで、zero のｚの文字がオプションにきています。なお、行列表示の際には、行列のまとまりの前に１行挿入されるというのがデフォルトになっています。

ポイント　行列の各行の要素間には、デフォルトとして、１スペースの間隔が存在する

　このように、１行の要素と要素の間のスペースはデフォルトで決まっているのですが、この１スペース分を変更することも可能です。具体的には、タブのスペース分だけあけるもの（ｔ）、１スペース分のかわりにコンマをつけるもの（ｃ）、そして１スペースの間をなくしてしまって間隔をなくするもの（ｎ）など、各種設定が可能です。

プログラム

```
new; cls;
rndseed 999;
a = rndn(3,3);
print "----|----1----|----2----|----3----|----4----|----5";
print "----|----|----|----|----|----|----|----|----|----|";
print /re a;
print /ret a;
print /rec a;
print /ren a;
print "----|----|----|----|----|----|----|----|----|----|";
```

画面結果

```
----|----1----|----2----|----3----|----4----|----5
----|----|----|----|----|----|----|----|----|----|

 1.32205752e+000   1.33155183e+000  -4.58844377e-001
 1.03048047e+000  -1.00846115e-001  -2.84676596e-001
 5.13184990e-001   1.17451874e+000   1.94213898e+000

 1.32205752e+000     1.33155183e+000    -4.58844377e-001
 1.03048047e+000    -1.00846115e-001    -2.84676596e-001
 5.13184990e-001     1.17451874e+000     1.94213898e+000

 1.32205752e+000, 1.33155183e+000,-4.58844377e-001,
 1.03048047e+000,-1.00846115e-001,-2.84676596e-001,
 5.13184990e-001, 1.17451874e+000, 1.94213898e+000,

 1.32205752e+000 1.33155183e+000-4.58844377e-001
 1.03048047e+000-1.00846115e-001-2.84676596e-001
 5.13184990e-001 1.17451874e+000 1.94213898e+000
----|----|----|----|----|----|----|----|----|----|
```

まず、最初の行列表示はデフォルトのものです。デフォルトでは、まず１行間隔をあけてから、行列を表示します。各行の要素と要素の間には １ スペースの空白が入ります。実際には、/res としても同じ結果となります。スペースを意味するオプションｓがこの場合省略されているのです。第２番目の行列表示は、タブ分の３スペース分ずつ間隔を空けるｔのオプションを/re のあとにつけています。その他のオプションの場合も同様に最後にｔをつけるとタブ間隔になります。ただし、タブの場合、ワープロなどに結果をペーストすると間隔がなくなっていまうことがあるので注意が必要です。第３番目の行列表示は、１スペースのかわりにカンマをつけるオプションです。ｃのオプションを/re のあとにつけています。このカンマは出力上の便宜的なカンマであって、それぞれが次の行になるというGAUSS の行列の規則はこの場合適用されません。あくまでも出力上のみかけのものです。第４番目のオプションは、スペースなしのｎオプションの場合です。これらのオプションは前者の２文字のオプションに加えて、あくまでスラッシュ付きの３文字として構成されます。ですから１文字だけスラッシュをつけても動作をしませんので注意してください。

ポイント　行列の各行の要素間の設定を変更するには

print /res 変数;　１スペースあける（デフォルト）

print /ret 変数;　タブ間隔をあける

print /rec 変数;　カンマで区切る

print /ren 変数;　間隔をあけない

さて、今度は横にではなくて、行列の各行の間を縦方向にあける方法です。引き続き re で説明していきます。上の場合同様、その他の桁数フォーマットでも同じようになります。

プログラム

```
new; cls;
rndseed 999;
a = rndn(3,3);
print "----|----1----|----2----|----3----|----4----|----5";
print "----|----|----|----|----|----|----|----|----|----|";
print /m0 /re a;
print /m1 /re a;
print /m2 /re a;
print /m3 /re a;
print "----|----|----|----|----|----|----|----|----|----|";
```

画面結果

```
----|----1----|----2----|----3----|----4----|----5
----|----|----|----|----|----|----|----|----|----|
 1.32205752e+000   1.33155183e+000  -4.58844377e-001   1.03048047e+000 .........

 1.32205752e+000   1.33155183e+000  -4.58844377e-001
 1.03048047e+000  -1.00846115e-001  -2.84676596e-001
 5.13184990e-001   1.17451874e+000   1.94213898e+000



 1.32205752e+000   1.33155183e+000  -4.58844377e-001

 1.03048047e+000  -1.00846115e-001  -2.84676596e-001

 5.13184990e-001   1.17451874e+000   1.94213898e+000

Row 1
 1.32205752e+000   1.33155183e+000  -4.58844377e-001
Row 2
 1.03048047e+000  -1.00846115e-001  -2.84676596e-001
Row 3
 5.13184990e-001   1.17451874e+000   1.94213898e+000
----|----|----|----|----|----|----|----|----|----|
```

今度は新しいオプションをまたもう１つ付け加えるという形をとります。/m0 とか/m1 というのは、/mb0 や/mb1 の略で、行列の前（before）に何かする意味でｂとなっています。通常は、ｂは省略されて、ｍのあとにすぐに数字がきます。最初の/m0 のオプションは、行列の各行の前にも後にも改行などしないでベタ出力をするというものです。ｍは matrix を表すｍで、０は０回改行するという意味です。この場合、デフォルトである行列のかたまりの前に１行入れることすら実行されません。２番目の/m1 オプションは、行列を通常どおり行列のそれぞれの行の前で１回改行すると言う意味でｍ１となっています。この場合には、１番最初の行の前にも１回改行が入るという意味で、デフォルトどおり行列のかたまりの前に１行入ります。３番目の/m2 オプションは、それぞれの行の前で２回改行するという意味です。ですから、最初の行の前には２回の改行の結果２行間隔があき、２行目以降の前には２回改行するのですから１行だけ間隔があきます。最後の/m3 オプションは、３回改行

するという意味ではなくて、ｍ２オプションのままの間隔で、各行の前に行（row）番号を挿入するという意味です。なお、前という意味のｂの反対は、後という意味のａもあります。すでに m0 があるので ma0 は存在しません。また、ma3 も存在しません。下に行ラベルをつけること自体が意味をなさないためであろうと思われます。

プログラム

```
new; cls;
rndseed 999;
a = rndn(3,3);
print "----|----1----|----2----|----3----|----4----|----5";
print "----|----|----|----|----|----|----|----|----|----|";
print /ma1 /re a;
print /ma2 /re a;
print "----|----|----|----|----|----|----|----|----|----|";
```

結果表示

```
----|----1----|----2----|----3----|----4----|----5
----|----|----|----|----|----|----|----|----|----|
 1.32205752e+000   1.33155183e+000  -4.58844377e-001
 1.03048047e+000  -1.00846115e-001  -2.84676596e-001
 5.13184990e-001   1.17451874e+000   1.94213898e+000

 1.32205752e+000   1.33155183e+000  -4.58844377e-001

 1.03048047e+000  -1.00846115e-001  -2.84676596e-001

 5.13184990e-001   1.17451874e+000   1.94213898e+000



----|----|----|----|----|----|----|----|----|----|
```

１番目の/ma1 は、各行の後（after）という意味でａがきていて、各行の後に改行を１回します。結果は、/m1 とほとんど同じですが、最初の行の前には１行入りません。なぜなら行の前ではなくて後に改行するからです。その代わり、行列のかたまりの後にかわりに１行入ります。２番目の/ma2 は、各行の後に２回改行するという意味で、各列の間には１行間

隔が入っています。ただし行のあとには２回改行するという意味で２行間隔があきます。なお、/m1 というグループと/re というグループは、前後どちらを先に書いてもかまいませんし、もちろん、片方だけ書いてもう片方を省略することもできます。通常は、/m1 のグループが先にきて、/re のグループを後に書かかれます。

ポイント　行列の縦方向の間隔を調整するのには

print /m0 変数;　各行の前に（後にも）改行を入れない　ベタ出力
print /m1 変数;　各行の前に１つ改行を入れる　１行挿入後飛ばしなし
print /m2 変数;　各行の前に２つ改行を入れる　２行挿入後１行飛ばし
print /m3 変数;　各行の前に２つ改行を入れて、行番号を直前に表示
print /ma1 変数;　各行の後に１つ改行を入れる　飛ばしなし後１行挿入
print /ma2 変数;　各行の後に２つ改行を入れる　１行飛ばし後２行挿入

以上が print 文だけで出力を制御する方法でしたが、このほかに format 文を print 文の前に書いて本格的に出力を整えることもできます。format 文は、いままで説明してきた print 文のすべてのオプションを同じようにスラッシュをつけて書くことができます。ただし、format 文にしかできないこともあります。それは全体の文字スペースの桁数と小数点以下の桁数を変えることができる点にあります。format 文を書かない場合の桁数のデフォルトは、全体の文字スペースの入る領域の桁数は 16 桁で、小数点以下は８桁となっています。そこで、このデフォルトを具体的に変更してみましょう。

プログラム

```
new; cls;
rndseed 999;
a = rndn(3,3);
print "----|----1----|----2----|----3----|----4----|----5";
print "----|----|----|----|----|----|----|----|----|----|";
format /re 12,4;
print a;
print "----|----|----|----|----|----|----|----|----|----|";
```

画面結果

```
----|----1----|----2----|----3----|----4----|----5
----|----|----|----|----|----|----|----|----|----|

 1.3221e+000   1.3316e+000  -4.5884e-001
 1.0305e+000  -1.0085e-001  -2.8468e-001
 5.1318e-001   1.1745e+000   1.9421e+000
----|----|----|----|----|----|----|----|----|----|
```

上のように、もともとは 16 桁分と確保されていた領域が、/re 12,4 と指定することによって、12 桁になりました。同時に、小数点以下の桁数も４桁になっています。このように、最初の数字は全体の領域の桁数、その次の数字は小数点以下の桁数になっています。（ただし、最初のほうで述べたように例外は規則に従ってたくさんあります。）比較対照する統計計量パッケージソフトが小数点以下４桁を出力するのであれば４を使えばよいでしょうし、５桁目が四捨五入されているのですが、そのくわしい状況が見たければ、その値を少し大きくしてやればよいでしょう。ただし、最初の 12 のところは、12 未満の数字にしてしまうと、桁数がしぼられてしまって非実用的です。また、小数点以下の数を必ず出力することを優先して、全体の桁数をたとえ８にしようと、８になるわけでもありません。注意が必要です。通常は、16 桁または 12 桁がよく使われるようです。小数点以下は、４桁から５桁で、比較対照される統計計量パッケージソフトの桁数によります。なお、デフォルト値は

format /mb1 /ros 16,8

16 桁は確保されますが、最初に述べたように ro の場合、小数点以下８桁に常になるわけではありません。rd がよりコンパクトと判断して選択された場合には、符号や小数点を除く正味８桁が出力されます。小数点の数を常に固定したい場合には/re のオプションで数値を指定する必要があります。整数だけを問題にする場合には、小数点以下を０にして、

format /rds 1,0

とするのが普通です。（ｓはデフォルトなので、つけてもつけなくてもかまわない。）１のカンマの前の数字は任意です。１であっても２であっても、十進法では全体の桁数の領域確保に関係なく、小数点以上の部分の数はあるだけ桁数は表示されます。

２つほど注意をうながしておきたいことがあります。まず１つ目は、format の桁数制御は非常に強い命令です。次に書きなおしたプログラムに、たとえ format コマンドがなくても、そのプログラムのデフォルトは前のプログラムの format の設定のまま残ってしまうことが多々あります。そんな場合には、1 度 GUASS を終了させて再び起動しなおす必要があります。もう１つは、print 文と同じように、format 文は print される出力の見かけ上の制御を行なうだけであって、計算の過程の有効数字をコントロールするものではありません。この点は、アメリカの偉い学者でも勘違いしている方がおられます。そういう勘違いをプログラム上で防止するために、format 文は print 文から離れてプログラムの最初の方に書くのではなくて、print 文の直前にいちいち書いてやると、プログラムを読んだり移植したりしている人の理解の助けになります。例えば、

format /m3 /re 16,4; print a;

と 1 行に続けて書いて、print の直前に format を置いてやるとわかりやすくなります。すこしだけプログラム例をあげておきましょう。基本的に、数値による桁数制御以外は、以前に述べた print 文単体の記述方法と同じです。print 文の記述をなくしてしまって、それを別個に format 文に書いて前においてやれば、簡単に出力を制御できます。なお、format 文と print 文にはそれぞれ別個の命令なので、それぞれ；のマークが必要です。

プログラム

```
new; cls;
rndseed 999;
a = rndn(3,3);
print "----|----1----|----2----|----3----|----4----|----5";
print "----|----|----|----|----|----|----|----|----|----|";
print a;
format /mb1 /ros 16,8; print a;
format /m3 /re 16,4; print a;
print "----|----|----|----|----|----|----|----|----|----|";
```

画面結果

```
----|----1----|----2----|----3----|----4----|----5
----|----|----|----|----|----|----|----|----|----|

        1.3220575         1.3315518       -0.45884438
        1.0304805       -0.10084612       -0.28467660
       0.51318499         1.1745187         1.9421390

        1.3220575         1.3315518       -0.45884438
        1.0304805       -0.10084612       -0.28467660
       0.51318499         1.1745187         1.9421390

Row 1
      1.3221e+000       1.3316e+000      -4.5884e-001
Row 2
      1.0305e+000      -1.0085e-001      -2.8468e-001
Row 3
      5.1318e-001       1.1745e+000       1.9421e+000
----|----|----|----|----|----|----|----|----|----|
```

以上のようにうまく出力されたでしょうか。１番目は、format をせずにそのまま出力したものです。２番目は、デフォルトの設定に format して出力したものです。当然のことながら、まったく同じ結果になります。最後の行列は、最初に２行とばして、各行の前に行番号を挿入して、全体として 16 桁確保して小数点以下４桁の指数表示で出力したものです。

ポイント　fromat 文は、具体的な桁数の設定が可能なほか、print 文の制御と同じオプションがつけられる。
例　format /re 12,4; print a;　12 桁確保、小数点以下４桁の指数表示

なお、format 文の/off オプションには、フォーマットを初期化したりデフォルトに戻したりする機能は残念ながらありません。例えば、/m3 オプションをつけたならば、仮にその後に別の format 文で出力を制御したとしても、/m 関係のオプションがない format 文では前回の format が GUASS を起動し直すまで効果は残ります。使いこなせれば、いかようにでも結果を飾るツールになりえますが、Format 制御は非常にセンシティブな命令なので十分に注意して使用することが肝要です。

最後にフォーマットの注意点と誤解について注意を促しておきます。

１）フォーマットは print 文を制御するだけで、途中計算の有効数字とは無関係である。
２）フォーマットの数値 **x,y** のところは、最初の数字が最低限確保される文字スペース、その次の数字が ｅ オプション時の小数点以下の桁数、ｄオプション時の全体の正味桁数（マイナスサインや小数点を除く）。ｏオプションは、それらのうちでよりコンパクトな表示を選択するもの。
３）ｒオプション以外の場合、ｙのところの数字が必ずしも小数点以下の桁数をさすものではない。また、ｄオプションの場合、ｘのところの数字が必ずしも確保される文字スペースの桁数と一致するものではない。それよりも長くなる可能性もある。
４）行列の縦の間隔をあけるためのオプションｍの後の数字は、各行の一番あとの（あるいは前）数字からの改行の数であって、行間隔をさしているわけではない。